Tornado: An Autonomous Chaotic Algorithm for Large Scale Global Optimization

Nassime Aslimani, El-Ghazali Talbi, Rachid Ellaia

Tornado: An Autonomous Chaotic Algorithm for Large Scale Global Optimization

N. Aslimani, E-G. Talbi, R. Ellaia[c]

[a]*University of Lille & Inria, France.*
[b]*University of Lille & Inria, France.*
[c]*Mohammed V University of Rabat, Morocco.*

Abstract

In this paper we propose an autonomous chaotic optimization algorithm, called Tornado, for large scale global optimization problems. The algorithm introduces advanced symmetrization, levelling and fine search strategies for an efficient and effective exploration of the search space and exploitation of the best found solutions. To our knowledge, this is the first accurate and fast autonomous chaotic algorithm solving large scale optimization problems.

A panel of various benchmark problems with different properties were used to assess the performance of the proposed chaotic algorithm. The obtained results has shown the scalability of the algorithm in contrast to chaotic optimization algorithms encountered in the literature. Moreover, in comparison with some state-of-the-art metaheuristics (e.g. evolutionary algorithms, swarm intelligence), the computational results revealed that the proposed Tornado algorithm is an effective and efficient optimization algorithm.

Keywords: Global optimization, Chaos optimization algorithm, Levelling, Symmetrization, Fine search, Large scale optimization.

1. Introduction

Chaos theory is a branch of mathematics dealing on the study of dynamical systems whose apparently-random states of disorder and irregularities are often governed by deterministic laws [1]. Chaotic behavior exists in many natural systems, including fluid flow, weather and climate. It also occurs spontaneously in some systems with artificial components, such as stock market and road traffic. Chaotic systems are characterized by high sensitivity to initial conditions, an effect which is popularly referred to as the butterfly effect [2]. As a result of this sensitivity, the behaviour of such systems appears to be stochastic, even though the model of the system is deterministic, meaning that their future behaviour is fully determined by their initial conditions, with no random elements involved. Another consequence of the butterfly effect is unpredictability. Small differences in initial input can have radically different results after several cycles through the system. In recent years, chaos has attracted widespread attention and have been widely applied in various disciplines such as control [3][4] and optimization [5].

Nowadays, there is a need for more effective and efficient optimization techniques, able to solve large scale problems with hundreds, thousands, and even millions of continuous variables. State-of-the-art chaos based optimization algorithms (COAs) are not efficient for large scale optimization problems [6]. They are not even operational for a dimension greater than 5 [6]. Existing COAs are deficient in terms of:

- *Exploration of the search space*: indeed, the irregularity of the chaos dynamics grows quickly with the problem dimension. This is due to the intrinsic imprevisibility of chaotic dynamics [6].
- *Exploitation of the best found solutions*: the main search mechanism used in existing COAs is not adapted for a good exploitation. It selects in a random way the direction around the current solution [7].

Email address: nassime.aslimani@inria.fr, el-ghazali.talbi@univ-lille.fr (N. Aslimani, E-G. Talbi, R. Ellaia)

This paper is the culmination of an approach that leads to an autonomous COA algorithm. First, the following strategies have been introduced in a gradient-based chaotic algorithm to improve the regularity and the flexibility of the algorithm [8]:

- *Symmetrization*: on the one hand, symmetrization induces a regular structure into the chaotic dynamics for a better exploration. On the other hand, based on a stochastic decomposition strategy, it enables an efficient and scalable alternative search mechanisms for a better exploitation in the search space.
- *Levelling*: in fact, the chaotic dynamics has been restructured by a leveling approach. This allows to generate different flexible chaotic levels to improve the exploration and the exploitation of the search space.
- *Hybridization with local search*: a combination with gradient based algorithm has been carried out for continuous differentiable functions.

In this paper, an autonomous Chaos is introduced which speedups the convergence and improves the accuracy of the search for large scale problems. An autonomous and pure chaotic algorithm has been developed, in which the combination with a local search algorithm (e.g. gradient descent) has been replaced by a chaotic fine search. The computational results for many test functions with different properties and levels of complexity has shown the effectiveness, efficiency, and scalability of the autonomous chaotic algorithm in tackling large scale optimization problems.

This paper is organized as follows. In section 2, the related work on chaos optimization algorithms and state-of-the art global optimization algorithms (e.g. evolutionary algorithms, swarm intelligence) are presented. Section 3 details the novel autonomous chaos optimization, the Tornado algorithm. Section 4 shows the computational experiments of the proposed algorithm. A comparison has been carried out as well with popular chaos optimization algorithms and state-of-the-art stochastic metaheuristics (e.g. evolutionary algorithms, swarm intelligence). The conclusion and the perspectives of this work are made in Section 5.

2. Related work

Consider an optimization problem with bounding constraints[1]:

$$\text{Minimize } f(X) \text{ subject to } X \in [L, U], \tag{1}$$

where

- $f : \mathbb{R}^n \longrightarrow \mathbb{R}$, denotes the objective function,
- $X = (x_1, .., x_n) \in \mathbb{R}^n$, the decision vector whose components x_i are bounded by lower bounds l_i and upper bounds u_i. and $[L, U] = \prod_{i=1}^{n} [l_i, u_i]$.

Chaos is a universal nonlinear phenomenon with stochastic, ergodic, and regular properties. Ergodicity can be used as a search mechanism for optimization. The sequence of solutions is generated by means of a chaotic map. Different chaotic maps exist in the literature [9]. The most popular ones are:

- The logistic map: $x_k = \mu.x_k(1 - x_k), \quad 0 < x_0 < 1, \quad 0 \leqslant \mu \leqslant 4$
- The Kent map: $x_{k+1} = \begin{cases} x_k/\beta & \text{if } 0 < x_k < \beta \\ (1 - x_k)/(1 - \beta) & \text{if } \beta < x_k < 1 \end{cases}$,
- The Henon map: $\begin{cases} x_{k+1} = a \times (1 - x_k^2) + b \times y_k \\ y_{k+1} = x_k \end{cases} \quad (x_0, y_0) \in \mathbb{R}^2, \ a, b > 0$

[1] without loss of generality, we consider only minimization problems.

Chaos has been embedded in the development of novel search strategies for global optimization known as chaos optimization algorithms (COAs). COAs have the properties of easy implementation, reduced execution time and robust mechanisms of escaping from local optimum. COA has been used in many applications such as optimization of power flow problems [10], control systems [11], neural networks [12], cryptography [13] and image processing [14]. In [9], the best chaotic sequences generated by sixteen different chaotic maps have been analyzed.

Chaos based optimization has been originally proposed in 1997 [5]. It includes generally two main stages:

- *Global search*: an exploration of the global search space is carried out. A sequence of chaotic solutions is generated using a chaotic map. Then, the objective functions are evaluated and the solution with the best objective function is chosen as the current solution.

- *Local search*: the current solution is assumed to be close to the global optimum after a given number of iterations, and it is viewed as the centre on which a little chaotic perturbation, and the global optimum is obtained through local search. The above two steps are iterated until some specified stopping criterion is satisfied.

Observations from existing COA algorithms reveal that COA still presents some drawbacks especially with problems involving large spaces. Furthermore, the exploration ability of the COA decreases with the increase of the dimension space particularly because of the irregularity and the rigidity of the chaos dynamic which does not always authorize the exploration of some isolated regions containing the global optimum. Moreover, chaotic search has poor fine search ability, and then existing COAs suffer from the exploitation aspect. Most of the efficient chaos based optimization algorithms (COAs), proposed in the literature, are hybrid algorithms. Used generally as a global search strategy, COA is combined with local search efficient procedures such as gradient descent [8], grey-wolf [15], golden section search [16], and stochastic metaheuristics (e.g. butterfly [17], particle swarm [18][19], cucko search [20], firefly [21], genetic algorithms [22]).

Hence, few articles proposed an autonomous COA algorithm for large scale global optimization [23][7][24]. Rather, COA has been widely involved in hybridization strategies, and by contrast, these few autonomous COA approaches involve only low-dimensional problems, and that reveals their limited efficiency and especially their incapacity in handling higher-dimensional problems [6]. According to the aforementioned difficulties, this paper presents a new autonomous COA approach based on new strategies including symmetrization, levelling, and fined local search.

In the last two decades, many efficient metaheuristics have been developed for tackling continuous optimization problems. Most of state-of-the-art algorithms are stochastic metaheuristics:

- *Differential evolution (DE)*: DE has two main control parameters that are required to be fixed by a user before the evolutionary process starts: the scaling factor F, and the crossover control parameter CR. Many adaptive and self-adaptive DE variants have been developed (e.g. L-SHADE [25]). jSO [26] and SHADE-cnEpSin [27] are DE-based winners of the CEC'2017 competition. SALSHADE-cnEpSin [28] and LSHADE-RSP [29] are the DE-based winners of the CEC'2018 competition.

- *Evolution strategies (ES)*: CMA-ES (Covariance Matrix Adaptation-Evolution Strategy) represents an efficient algorithm for global optimization [30]. CMA-ES is a population based multivariate sampling algorithm, in which new candidate solutions are sampled according to the multivariate normal distribution, which is implemented by using the adaptation of covariance matrix.

- *Particle swarm optimization (PSO)*: designing learning methods that can use previous search information more efficiently was one of the most salient PSO research topics. The Orthogonal Learning PSO (OLPSO) [31] and the heterogeneous CLPSO [32] represent one of the most efficient PSO-based algorithms to solve global optimization problems. In OLPSO, orthogonal learning (OL) strategy is used to discover useful information and guide particles to fly in better directions by constructing a much promising and efficient exemplar [31]. In CLPSO, the swarm population is divided into two subpopulations. Each subpopulation is assigned to focus solely on either exploration or exploitation. Comprehensive learning (CL) strategy is used to generate the exemplars for both subpopulations [32].

- *Estimation of distribution algorithms (EDA)*: the principle of EDA is to explore the space of potential solutions by generating and sampling promising solutions [33]. The main stage is the construction of an explicit probabilistic model that tries to capture the probability distribution of the promising solutions by using tree-structured or Bayesian networks [34]. As the univariate EDAs assume that all the variables are independent, it is widely used to solve separable problems [35]. It has been shown that univariate EDAs such as univariate marginal distribution algorithm continuous (UMDAc) is efficient for solving some multimodal nonseparable problems [36][37].

- *Hybrid metaheuristics*: the hybrid metaheuristic LSHADE_SPACMA (Semi-Parameter Adaptation Hybrid with CMA-ES) shows its efficiency for solving the CEC'2017 benchmark problems [38]. The HS-ES (Hybrid Sampling Evolution Strategy) is the general winner of the CEC'2018 competition on real parameter bound-constrained optimization [39]. It combines CMA-ES and univariate sampling UMDAc algorithms. Univariate sampling is very effective for solving multimodal nonseparable problems. As the CMA-ES has obvious advantages for solving unimodal nonseparable problems, the proposed HS-ES tries to take advantages of these two complementary algorithms to improve the performance of the search.

3. The Tornado algorithm

The proposed Tornado algorithm is composed of three main procedures:

- The chaotic global search (CGS): CGS is a full exploration-based Chaotic search procedure. Its goal is to produce initial solutions that will be improved and refined by other exploitation-based chaotic search procedures.

- The chaotic local search (CLS): CLS is an exploitation-based Chaotic search procedure. Starting from an initial solution given by CGS, it exploits the neighborhood of the solution. By focusing on successive promising solutions, CLS allows also the exploration of promising neighboring regions.

- The chaotic fine search (CFS): CFS is a full exploitation-based Chaotic procedure. It uses a coordinate adaptive zoom strategy to intensify the search around the current optimum.

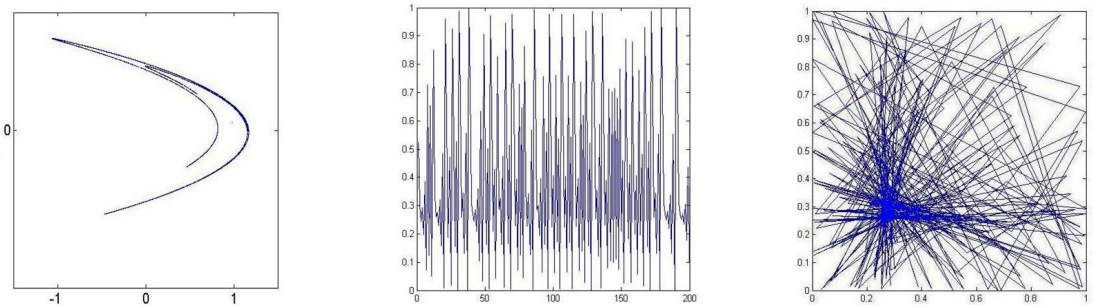
The structure of the proposed Tornado approach is given in Algorithm 1. In this work, we use the Henon map as a generator of a chaotic sequence. We consider a sequence $(Z_k)_{1\leq k\leq N_h}$ of normalized Henon vectors $Z_k = (z_{k,1}, z_{k,2}, .., z_{k,n}) \in \mathbb{R}^n$ through the following linear transformation of the standard Henon map (2) (Fig. 1):

$$z_{k,i} = \frac{y_{k,i} - \alpha_i}{\beta_i - \alpha_i}, \quad \forall\ (k,i) \in [\![1, N_h]\!] \times [\![1, n]\!], \tag{2}$$

where $\alpha_i = min_k(y_{k,i})$ and $\beta_i = Max_k(y_{k,i})$. Thus, we get $\forall\ (k,i) \in [\![1, N_h]\!] \times [\![1, n]\!], \quad 0 \leqslant z_{k,i} \leqslant 1$. In this work, the sequence of normalized Henon map vectors (Z_k) is defined as: $a = 1.5, \quad b = 0.2, \quad \forall k \in [\![1, n]\!], \quad (x_{k,0}, y_{k,0}) = (r_k, 0),\ r_k \sim U(0,1)$.

Algorithm 1 : The Tornado algorithm structure

```
Initialisation of the Henon chaotic sequence ;
Set k = 1 ;
Repeat
    Chaotic Global Search (CGS);
    Set s = 1 ;
    Repeat;
        Chaotic Local Search (CLS);
        Chaotic Finest Search (CFS);
        s = s + 1;
    Until s = M_l ; /* M_l is the number of CLS/CFS by cycle */
    k = k + 1 ;
Until k = M ; /* M is maximum number of cycles of Tornado */
```



(a) *Henon map attractor* (b) *Henon map (200 iterations)* (c) *Henon map dynamic in 2D (200 iterations)*

Figure 1: Illustration of Henon Map.

In general, chaos dynamics suffer from irregularity and rigidity, which induces a deficient exploration [9]. Indeed, the chaos dynamics does not always enable to cover some isolated regions of the search space. For a better exploration of the search space, the proposd *Tornado* algorithm uses symmetrization and levelling strategies to better control the flexibility and the orientation of the chaotic search process. In fact, those proposed strategies provide a distribution of chaos variables that contains several layers of symmetric solutions.

3.1. Chaotic global search (CGS)

CGS uses the following standard transformation to map chaos variable Z into ranges of design variable X_1:

$$X_1 = L + Z(U - L). \tag{3}$$

In addition, CGS uses two alternative transformations to improve the exploration capability of the decision space and then remedy the lack of exploration of the dynamic sequence given by the standard transformation (eq.3) (Fig. 2):

$$X_2 = \theta + Z(U - \theta), \quad X_3 = U - Z(U - \theta), \quad \theta = \frac{1}{2}(L + U). \tag{4}$$

Since $\forall i \in [\![1, n]\!]$: $0 \leqslant z_i \leqslant 1$ and $u_i - \theta_i = u_i - \frac{l_i+u_i}{2} = \frac{1}{2}(u_i - l_i) \geqslant 0$. Then, $0 \leqslant z_i(u_i - \theta_i) \leqslant u_i - \theta_i$ and $-u_i + \theta_i \leqslant -z_i(u_i - \theta_i) \leqslant 0$. Hence, $\theta_i \leqslant \theta_i + z_i(u_i - \theta_i) \leqslant \theta_i + u_i - \theta_i$ and $-u_i + \theta_i + u_i \leqslant u_i - z_i(u_i - \theta_i) \leqslant u_i$. In other words, $\theta_i \leqslant \theta_i + z_i(u_i - \theta_i) \leqslant u_i$ and $\theta_i \leqslant \theta_i - z_i(u_i - \theta_i) \leqslant u_i$. Thus, $X_2,\ X_3 \in [\theta, U] \subset [L, U]$.

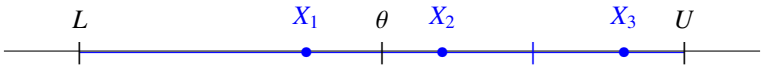
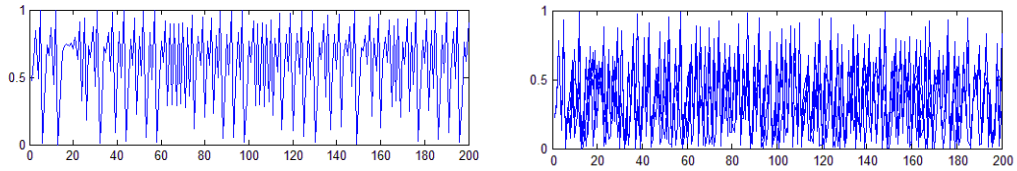


Figure 2: Selection of chaotic variables for CGS.

Those proposed transformations break the inherent rigidity of the chaos dynamics. Furthermore, the presence of the attractor in the chaos dynamic involves an implicit correlation between certain components of the chaos vector X because they are all ruled by the intrinsic law of the same involved chaos. To prevent the higher correlation that reduces significantly the ergodicity and consequently the exploration abilities of these chaotic transformations, we propose doubling the chaos effect by superposing a second chaos layer to the original chaos vector, that is by considering $Z_l Z_k$ instead of Z_k (Fig.3).



(a) *chaos vector Z_k for k=1..200 iterations*

(b) *average of double chaos vectors $(Z_l Z_k)_{1 \leqslant l \leqslant 5}$ for k=1..200 iterations*

Figure 3: Illustration of double chaos effect.

This mechanism involves a restructuring of the chaos dynamic by a levelling approach. Each chaos dynamic given by Z_l provides a new chaos level.

- *Levelling approach:* the CGS procedure generates three chaotic variables for each iteration k, and in each level, $l \in [\![1, N_c]\!]$ according to Figure2:

$$X_1 = L + Z_l Z_k \times (U - L) \tag{5}$$
$$X_2 = \theta + Z_l Z_k \times (U - \theta) \tag{6}$$
$$X_3 = U - Z_l Z_k \times (U - \theta). \tag{7}$$

  Note that we drop k from the subscript in the notation $X_{i,k}$ for sake of simplicity.

- *Symmetrization approach:* the exploration of all the dimensions in a high-dimensional space is not practical because of combinatorial explosion. Instead of that, we have introduced a new strategy consisting of a stochastic decomposition of the search space $\mathbb{R}^n$ into two vectorial subspaces: a vectorial line $\mathcal{D}$ and its corresponding hyperplane $\mathcal{H}$:

$$\mathbb{R}^n = \mathcal{D} \oplus \mathcal{H}, \quad \mathcal{D} = \mathbb{R} \times e_p, \quad \mathcal{H} = \text{vect}(e_i)_{i \neq p}. \tag{8}$$

  By consequence,

$$\forall\, X = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n: \quad X = X_d + X_h, \tag{9}$$

  where

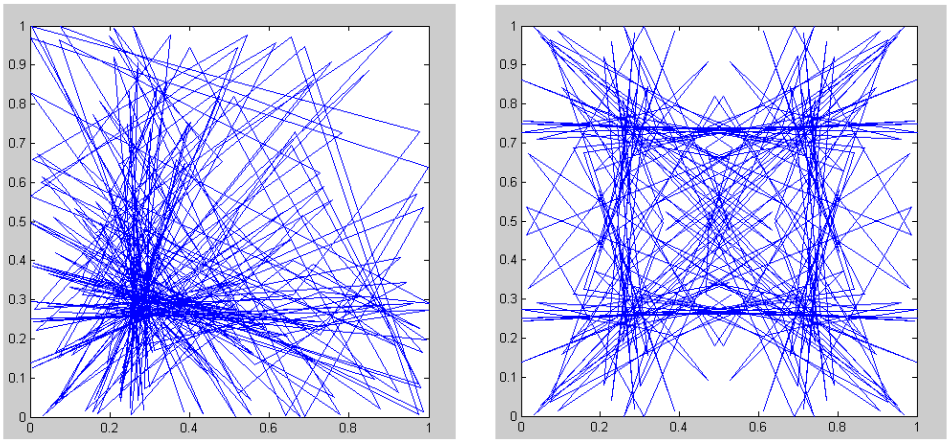
$$X_d = (0, \cdot\cdot, 0, x_p, 0, \cdot\cdot, 0) \in \mathcal{D}, X_h = (x_1, \cdot\cdot, x_{p-1}, 0, x_{p+1}, \cdot\cdot, x_n) \in \mathcal{H}. \tag{10}$$

The symmetrization approach based on this stochastic decomposition of the design space has two main consequences:

- Reduces significantly the complexity of the high dimensional problem in a way as if we were dealing with a 2D space with four directions.
- The symmetric chaos is more regular and more ergodic than the basic one (Fig. 4).

Consider the axial symmetries $\mathcal{S}_{\theta+\mathcal{D}}$, $\mathcal{S}_{\theta+\mathcal{H}}$ defined as follows:

$$\mathcal{S}_{\theta+\mathcal{D}}(X) = X_d + (2\theta_h - X_h) \tag{11}$$
$$\mathcal{S}_{\theta+\mathcal{H}}(X) = (2\theta_d - X_d) + X_h \tag{12}$$



(a) *Henon map dynamic in 2D (200 iterations)* (b) *Symmetrized Henon map (200 iterations)*

Figure 4: Illustration of symmetrisation approach in 2D.

Therefore, based on the stochastic decomposition (eq.8), in each chaotic level $l \in [\![1, N_c]\!]$, CGS generates four symmetric chaotic points using axial symmetries $\mathcal{S}_{\theta+\mathcal{D}}$, $\mathcal{S}_{\theta+\mathcal{H}}$ (Fig. 5):

$$\begin{array}{llll} X_{i,1} & = X_i, & X_{i,2} & = \mathcal{S}_{\theta+\mathcal{D}}(X_{i,1}), \\ X_{i,3} & = \mathcal{S}_{\theta+\mathcal{H}}(X_{i,2}), & X_{i,4} & = \mathcal{S}_{\theta+\mathcal{D}}(X_{i,3}) = \mathcal{S}_{\theta+\mathcal{H}}(X_{i,1}). \end{array} \tag{13}$$

In other words, $\forall i \in \{1, 2, 3\}$:

$$\begin{array}{llll} X_{i,1} & = X_i = X_{i,d} + X_{i,h}, & X_{i,2} & = X_{i,d} + 2\theta_h - X_{i,h}, \\ X_{i,3} & = 2\theta - X_{i,1}, & X_{i,4} & = 2\theta - X_{i,2}. \end{array} \tag{14}$$

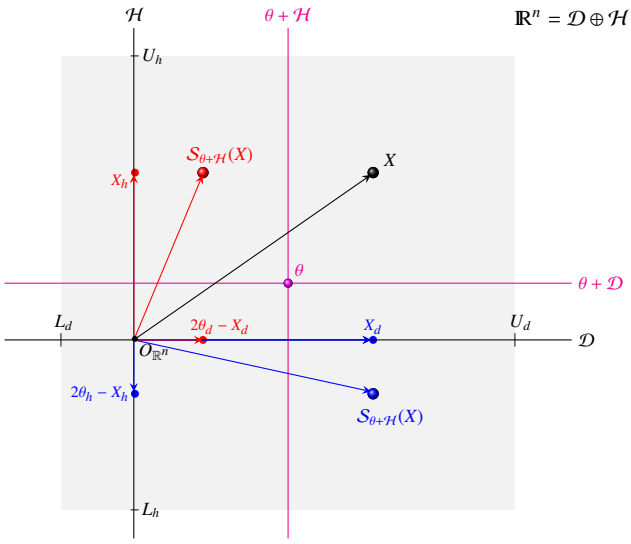


Figure 5: Illustration of axial symmetries $\mathcal{S}_{\theta+\mathcal{D}}$ and $\mathcal{S}_{\theta+\mathcal{H}}$.

At last, the best solution among these all generated chaotic points is selected as illustrated by Fig.6.

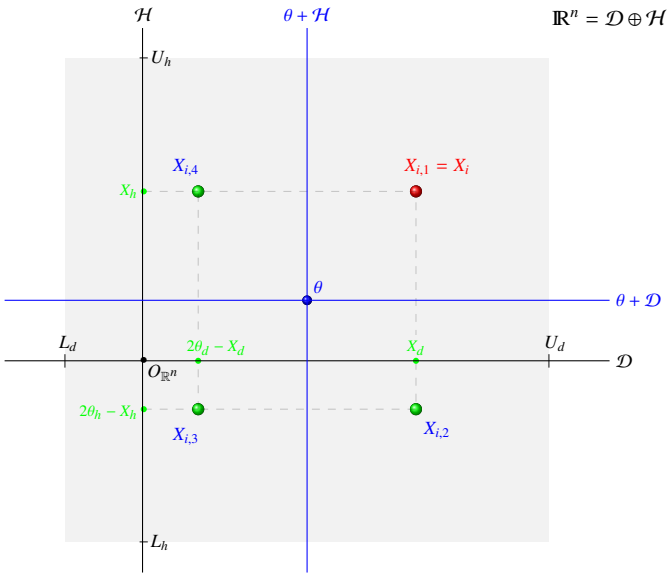


Figure 6: Generation of chaotic variables by the symmetrization approach in CGS.

Note that, in each chaotic level η of CGS, a new direction $\mathcal{D}$ is randomly selected with its corresponding hyperplane $\mathcal{H}$. This dynamic character of this stochastic decomposition through the search process is a key point for the efficiency of the CGS because this ensures the regularity, and the diversity of the chaotic dynamics, and as a consequence, it makes the exploration more equitable and so more efficient. The code of CGS is detailed in Algorithm 3.

Algorithm 2 Chaotic global search (CGS).

Input: f, U, Z, N_c, k
Output: X_c
$Y = +\infty$; $\theta = \frac{1}{2}(U + L)$
for $l = 1$ **to** N_c
  Generate three chaotic variables X_1, X_2, and X_3 according to the following:
  $X_1 = \theta + Z_l Z_k \times (U - \theta)$, $X_2 = U - Z_l Z_k \times (U - \theta)$, $X_3 = L + Z_l Z_k \times (U - L)$
  for $i = 1$ **to** 3
    Select **randomly** an index $p \in \{1, \cdots, n\}$ and **decompose** X_i according to (eq.9)
    Generate the four corresponding symmetric points $(X_{i,j})_{1 \leqslant j \leqslant 4}$ according to (eq.13) and (eq.14)
    for $j = 1$ **to** 4
      if $Y > f(X_{i,j})$
        $X_c = X_{i,j}$; $Y = f(X_{i,j})$
      end if
    end for
  end for
end for

3.2. Chaotic local search (CLS)

The CLS procedure allows to refine the search by exploiting the neighbourhood of the solution $\boldsymbol{\omega}$ found by the chaotic global search CGS. However, CLS contributes also to the exploration of the decision space by looking for potential solutions relatively far from the current solution. In fact, the CLS conducts the search process near the current solution ω within a local search area $\mathcal{S}_\mathbf{l}$ of radius $\mathcal{R}_\mathbf{l} = r \times \mathcal{R}$ focused on ω (see Fig.7a), where $r \sim U(0,1)$ is a random parameter corresponding to the reduction rate, and $\mathcal{R}$ denotes the radius of the search area $\mathcal{S} = \prod_{i=1}^{n} [l_i, u_i]$ defined as follows:

$$\mathcal{R} = \frac{1}{2}(U - L) = \left(\frac{1}{2}(u_1 - l_1), \ldots, \frac{1}{2}(u_n - l_n)\right) \tag{15}$$

The CLS procedure uses the following strategy to produce chaotic variables:

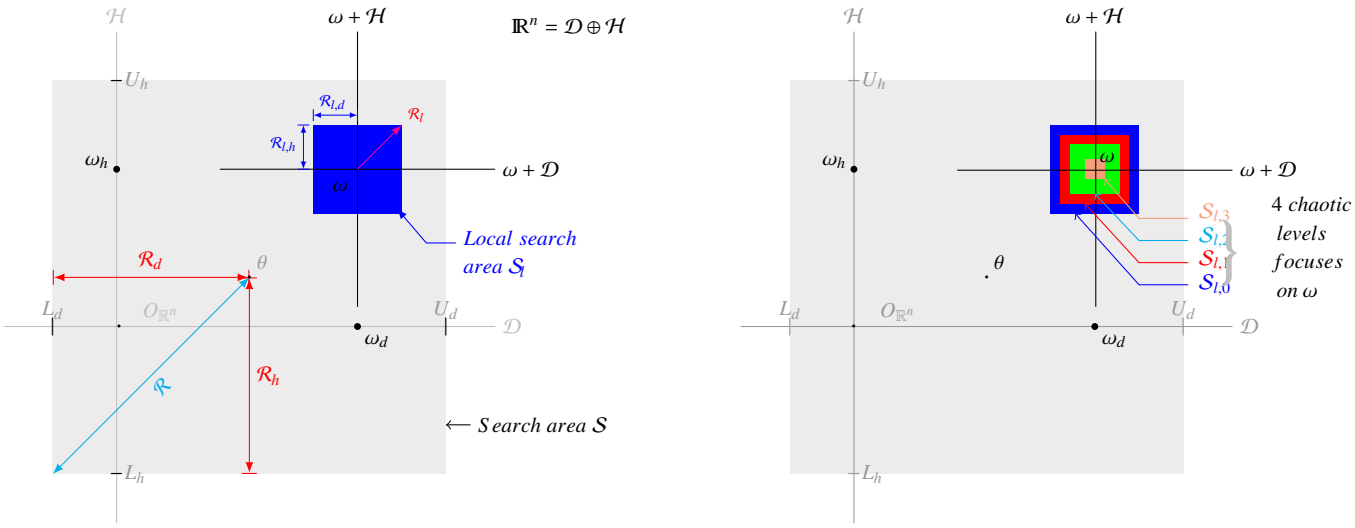
- Like the CGS, the CLS also uses a levelling approach by creating $N_\mathbf{l}$ chaotic levels focused on ω. In each chaotic level $\eta \in [\![0, N_l - 1]\!]$, the local search process is limited to a local area $\mathcal{S}_{l,\eta}$ focused on ω (see Fig.7b) characterized by its radius $\mathcal{R}_\eta$ defined by the following:

$$\mathcal{R}_\eta = \gamma_\eta \times \mathcal{R}_l = r \times \gamma_\eta \times \mathcal{R}, \tag{16}$$

  where γ_η is a decreasing parameter trough levels which we have formulated in this work as follows:

$$\gamma_\eta = \frac{10^{-2r\eta}}{1 + \eta} \tag{17}$$

where $r \sim U(0,1)$ is a random number distributed uniformly within the range $[0, 1]$.



(a): Illustration of local search area $\mathcal{S}_l$ of radius $\mathcal{R}_l$

(b) Zoom on current solution ω using the levelling approach

Figure 7: Illustration of the CLS mechanism.

In fact, the levelling approach used by the CLS corresponds to a progressive zoom focus on the current solution ω carried out through $N_\mathbf{l}$ chaotic levels, and γ_η is the factor (decreasing throughout the chaotic levels η) that controls the speed of this zoom process ($\gamma_\eta \searrow 0$). The aim is to look for potential optimal points in the area delimited by $\mathcal{R}_l$ but with much interest for the immediate neighbourhood of the current solution ω (see Fig.7b).

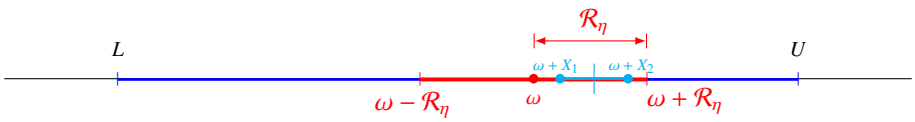


Figure 8: Selection of symmetric chaotic variables in CLS.

Indeed, once the CGS provides an initial solution ω, the CLS intensifies the search around this solution, through several chaotic layers. In each cycle of the Tornado algorithm, a given number (i.e. M_l) of CLS procedures is applied. Hence, the CLS participates also to the exploration of neighboring regions by following the zoom dynamic as shown in Fig.9.

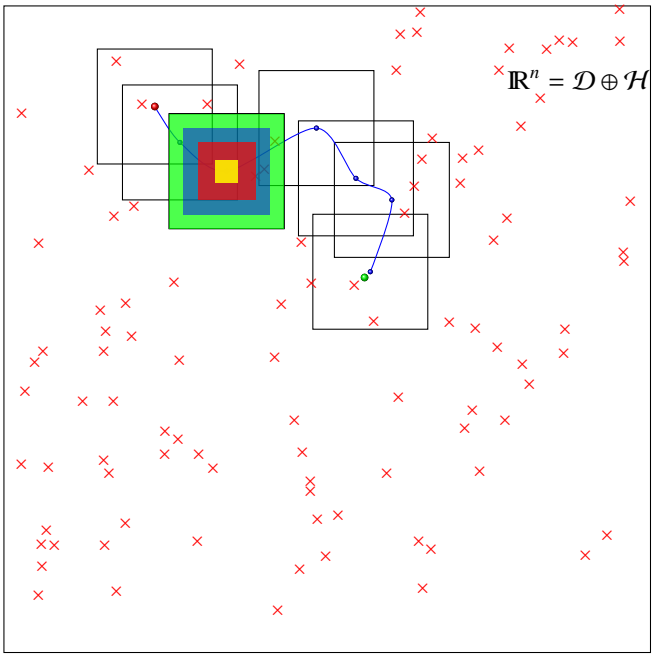


Figure 9: Illustration of the selection of symmetric chaotic variables in CLS.

Moreover, in each chaotic level η, CLS generates two symmetric chaotic variables X_1, X_2 according to Figure 8:

$$X_1 = Z \times \mathcal{R}_\eta, \quad X_2 = (1 - Z) \times \mathcal{R}_\eta = \mathcal{R}_\eta - X_1 \ . \tag{18}$$

We select randomly an index $p \in \{1, .., n\}$ and generate the corresponding stochastic decomposition of $\mathbb{R}^n$:

$$\mathbb{R}^n = \mathcal{D} \oplus \mathcal{H}, \quad \mathcal{D} = \mathbb{R} \times e_p, \quad \mathcal{H} = \text{vect}(e_i)_{i \neq p}. \tag{19}$$

Then, we get the corresponding decomposition of each chaotic variable $X_{i,(i=1,2)}$:

$$X_i = X_{i,d} + X_{i,h}. \tag{20}$$

Finally, we generate from each chaotic variable $X_{i,(i=1,2)}$, N_p symmetric chaotic points $(X_{i,j})_{1 \leqslant j \leqslant N_p}$ using the polygonal model (Fig.10):

$$X_{i,j} \quad = \omega + X_i = \omega + \cos(2\pi.j/N_p) X_{i,d} + \sin(2\pi.j/N_p) X_{i,h}, \tag{21}$$

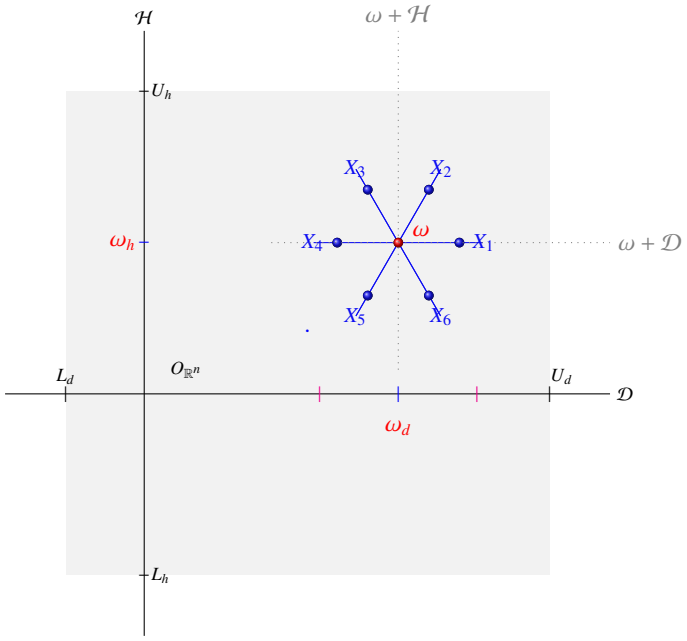


Figure 10: Illustration of the generation of $N_p = 6$ symmetric chaotic points in CLS.

Moreover, the local area selection process by reduction (via reduction factor r) requires some precautions in order to avoid the "sideeffect", because if ω is close enough to the borders of the search area $\mathcal{S}$, the search process risks to leave it and then it may give an infeasible solution localized outside $\mathcal{S}$.

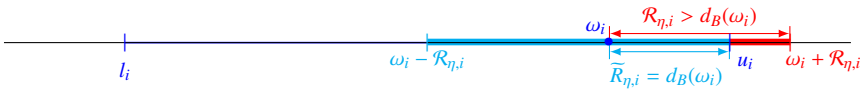


Figure 11: Illustration of overflow:$\mathcal{R}_{\eta,i} > d_B(\omega_i)$.

In fact, that occurs when $\mathcal{R}_{\eta,i} > d_B(\omega_i)$ for at least one component ω_i (Fig.11), where $d_{\mathrm{B}}(\omega_i)$ denotes the distance of the component ω_i to borders l_i, u_i defined as follows:

$$d_{\mathrm{B}}(\omega_i) = min(u_i - \omega_i, \omega_i - l_i). \tag{22}$$

To prevent this overflow, we consider the improved radius $\widetilde{R}_\eta$ instead of $\mathcal{R}_\eta$, given by the following:

$$\widetilde{R}_\eta = min(\mathcal{R}_\eta, d_{\mathrm{B}}(\omega)\underline{i}g), \tag{23}$$

where $d_{\mathrm{B}}(\omega) = (d_{\mathrm{B}}(\omega_1), \ldots, d_{\mathrm{B}}(\omega_n))$. This guarantees $\widetilde{R}_{\eta,i} \leqslant d_{\mathrm{B}}(\omega_i)$, $\forall i \in [\![1, n]\!]$. Hence, equations (18) become

$$X_1 = Z \times \widetilde{R}_\eta, \quad X_2 = (1 - Z) \times \widetilde{R}_\eta. \tag{24}$$

Finally, the algorithm of the chaotic local search (CLS) is described in Algorithm 3.

3.3. Chaotic fine search (CFS)

Chaotic search has limited fine search ability. The proposed CFS procedure allows to speed up the intensification process and refines the accuracy of the search. Suppose that the solution X obtained by the method CLS is close to the global optimum X_o with precision 10^{-p}, $p \in \boldsymbol{N}$. Then, we have:

$$X = X_o + \varepsilon, \quad \|\varepsilon\| < 10^{-p} \tag{25}$$

Algorithm 3 : Chaotic Local Search (CLS)

Input: f, ω, L, U, Z, N_l, N_p
Output: X_l: *best solution among the local chaotic points*
$\mathcal{R} = \frac{1}{2}(U - L)$; $\mathcal{R}_l = r \times \mathcal{R}$;
$X = \omega$;
$X_l = \omega$; $Y = f(\omega)$;
for $\eta = 0$ **to** $N_l - 1$
  Set $\mathcal{R}_\eta = \gamma_\eta \times \mathcal{R}_l$, and then compute $\widetilde{R}_\eta = min(\mathcal{R}_\eta, d_B(\omega))$
  Generate 2 symmetric chaotic variables X_1, X_2 according to (24)
  for $i = 1$ **to** 2
    Select an index $p \in \{1, .., n\}$ **randomly** and **decompose** X_i according to (20)
    Generate the N_p corresponding symmetric points $X_{i,j}$ according to (21)
      for $j = 1$ **to** N_p
        if $Y > f(X_{i,j})$ **then**
          $X_l = X_{i,j}$; $Y = f(X_{i,j})$;
        end if
      end for
    end for
end for

Thus, the distance ε can be interpreted as a parasitic signal of the solution, which is sufficient to filter in a suitable way to reach the global optimum, or the distance to which is the global optimum of its approximate solution. We carry out a chaotic search in a local area in which the radius adapts to the distance $\varepsilon = X - X_o$, component by component. However, in practice, the global optimum is not known a priori. To work around this difficulty, knowing that as the search process proceeds the resulting solution X is supposed to be close enough to the overall optimum, the trick found is to consider instead of the relation (25) the difference between the current solution X and its decimals fractional parts of order η, ($\eta \in \boldsymbol{N}$):

$$\varepsilon_\eta = |X - X_\eta|$$

where the fractional of order η, X_η is the closest point of X to the precision $10^{-\eta}$ defined by: $X_\eta = 10^{-\eta} round(10^\eta X)$.

For instance, Table 1 illustrates the fifth fractional parts as well as the corresponding errors for $X = (2.854732, 1.384527)$.

Order k	$k-$ Fractional part	Error of order k
0	$X_0 = (3, 1)$	$\varepsilon_0 = (0.145268, 0.384127)$
1	$X_1 = (2.9, 1.4)$	$\varepsilon_1 = (0.045268, 0.084127)$
2	$X_2 = (2.85, 1.38)$	$\varepsilon_2 = (0.004732, 0.004127)$
3	$X_3 = (2.855, 1.384)$	$\varepsilon_3 = (0.000268, 0.000127)$
4	$X_4 = (2.8547, 1.3841)$	$\varepsilon_4 = (0.000032, 0.000027)$

Table 1: Illustration of 5 first fractionnal and their corresponding errors.

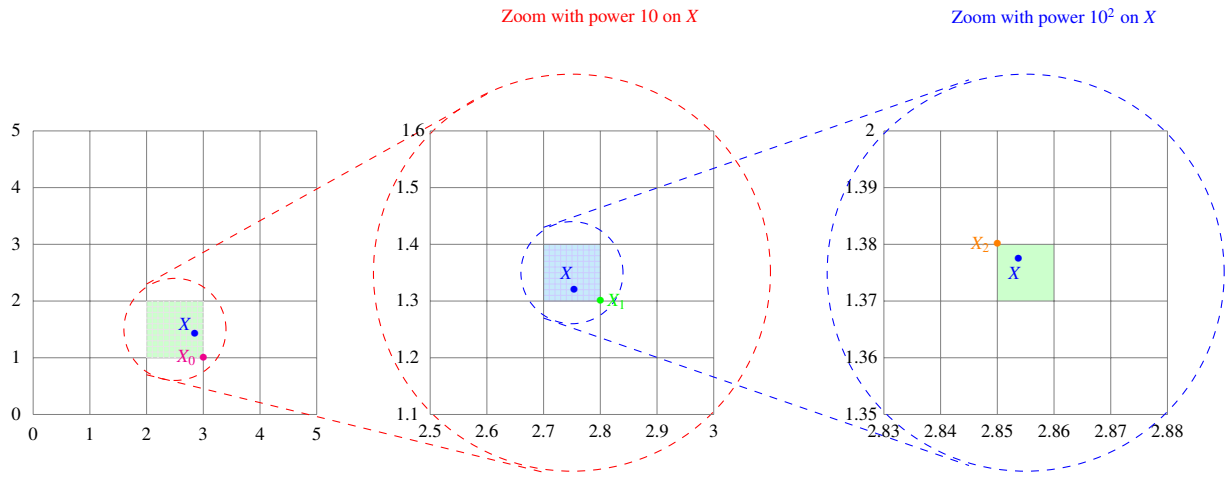


Figure 12: Illustration of the 10 power zoom via the successive fractional parts.

Moreover, in order to perturb a potential local optima we propose to add a stochastic component in the round process, in fact we consider the stochastic round $[.]_{st}$ formalised as:

$$[X]_{st} = \begin{cases} round(X) + P & , if \quad mod(k,2) = 0 \\ round(X) & , otherwise \end{cases} \tag{26}$$

where $P \sim U(-1,1)^d$ is a stochastic perturbation operated on X alternatively during the process. Thus, we get a new formulation of the $\eta-$error of X:

$$\widetilde{\varepsilon}_\eta(X) = |X - 10^{-\eta}[10^\eta X]_{st})| \tag{27}$$

The chaotic fine search CFS has a structure similar to the CLS local chaotic search. Indeed it operates by levelling on N_f levels, except the fact that the local area of level η is defined by its radius $\mathcal{R}_\eta$ proportional to the $\eta-$error ε_η and given by:

$$\mathcal{R}_\eta = \frac{1}{1+\eta^2}\widetilde{R}, \quad \eta \in [\![0, N_f - 1]\!] \tag{28}$$

This way the local area search is carried out in a narrow domain that allow a focus adapted coordinate by coordinate unlike the uniform local search in CLS. The modified radius $\widetilde{R}$ is defined by the following:

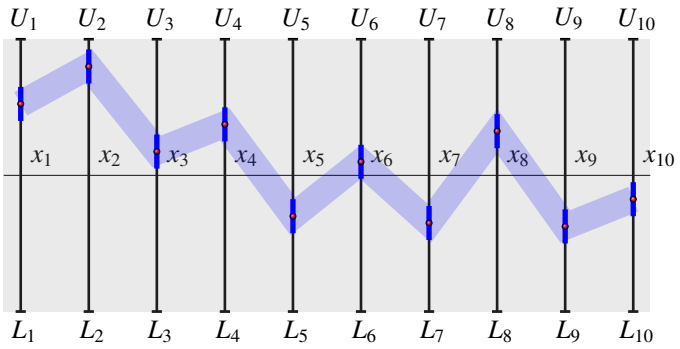
$$\widetilde{R} = \begin{cases} s \times R \cdot \widetilde{\varepsilon}_\eta & , if \quad r > 0.5 \\ T \cdot R \cdot \widetilde{\varepsilon}_\eta & , otherwise \end{cases} \tag{29}$$

where $r, s \sim U(0,1)$ and $T \sim U(0,1)^d$.

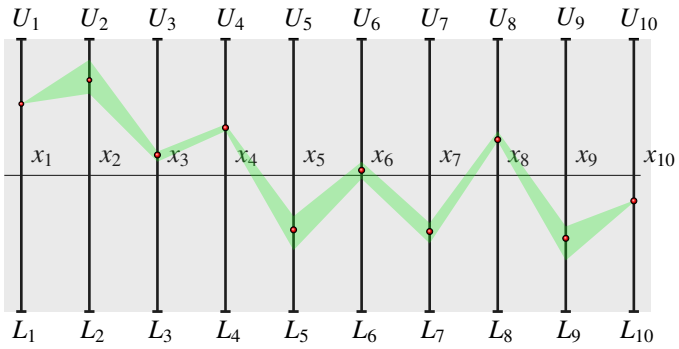
The $\mathcal{R}_\eta$ radius design allows to zoom at an exponential rate of decimale over the levels. Indeed, we have:

$$\|\mathcal{R}_\eta\| \leqslant \|\varepsilon_\eta\|.\mathcal{R} < 10^{-\eta} \times \mathcal{R} \tag{30}$$

Thus, the fine chaotic search allows an ultra fast exploitation of the immediate neighbourhood of the current solution and allows in principle the refinement of the global optimum with a good precision.



(a) Illustration of the uniform local search area in CLS using uniform reduction factor



(b) Illustration of the coordinate adaptative local search area in CFS based on the fractionnal error information ε_η

Figure 13: Illustration of the coordinate adaptative local search in CFS.

The Fine Chaotic Search (CFS) algorithm is the following:

Algorithm 4 : Chaotic Fine Search (CFS)

Input: f, ω, L, U, Z, N_f, N_p
Output: X_l: *the best solution among local chaotic points*
$\mathcal{R} = \frac{1}{2}(U - L)$;
$X = \omega$;
$X_l = \omega$; $Y = f(\omega)$;
for $\eta = 0$ **to** $N_l - 1$ **do**
  Compute the $\eta-$error $\widetilde{\varepsilon_\eta}$ and then evaluate $\widetilde{R}_\eta$ using equations (28)–(30)
  Generate two symmetrical chaotic variables X_1, X_2 according to (24)
  for $i = 1$ **to** 2
    Choose **randomly** p in $\{1, \cdots, n\}$ and **decompose** X_i using (20)
    Generate N_p symmetrical points $X_{i,j}$ according to (21)
    for $j = 1$ **à** N_p
      if $Y > f(X_{i,j})$ **then**
        $X_l = X_{i,j}$; $Y = f(X_{i,j})$;
      end if
    end for
  end for
end for

Finally the Tornado algorithm is detailed by the following algorithm:

Algorithm 5 Tornado Pseudo-Code.

Given : f, L, U, Z, M, M_l, N_c, N_l, N_f, N_p
Output : X, Y
$k = 1$; $Y = +\infty$;
while $k \leqslant M$ **do**
  $X_c =$ **CGS** (f, L, U, Z_k, N_c)
  if $Y > f(X_c)$ **then**
    $X = X_c$; $Y = f(X_c)$;
  end for
  $s = 1$;
  while $s \leqslant M_l$ **do**
    $X_l =$ **CLS** $(f, X, L, U, Z_{s+k}, N_l, N_p)$
    if $Y > f(X_l)$ **do**
      $X = X_l$; $Y = f(X_l)$;
    end if
    $X_f =$ **CFS** $(f, X, L, U, Z_{s+k}, N_f, N_p)$
    if $Y > f(X_l)$ **do**
      $X = X_f$; $Y = f(X_f)$;
    end if
    $s = s + 1$;
  end while
  $k = k + 1$;
fin tant que

4. Computational experiments

In this section, computational experiments are carried out in order to assess the performance of the proposed Tornado algorithm for large scale problems (i.e. 50, 100, and 200 variables). All the experiments were run using Intel(R) Core(TM) $i3$ 4005U CPU 1.70 GHz with 4 GB RAM. The implementation of all used algorithms was done in MatLab. Upon recommendation from CEC conference competitions[2], a set of 24 well known benchmark problems were selected with diverse properties and different levels of complexity (i.e. unimodal, multimodal, separable, non separable, shifted, rotated, noisy) as illustrated by Table 2. Unimodality shows the exploitation capability of the developed algorithms, while multi-modality confirms the exploration capabilities. The shifted global optimum for all the functions is provided as $o = (o_1, o_2, .., o_D)$ and the functions are defined as $z = x - o$ for shifted functions and $z = (x - o).M$ for shifted rotated functions where M is the transformation matrix for the rotating matrix. For instance, $F_1 - F_6$ are shifted functions and $F_7 - F_{12}$ are shifted and rotated functions.

[2]Competition on single objective real-parameter numerical optimization.

Table 2: High dimensional Benchmark functions used in our experiments.

B.Function	Expression	C	Search region	Optimum
Shifted Bent Cigar	$f_1 = y_1^2 + 10^6 \sum_{i=2}^{D} y_i^2 + bias,\ y = x - o,\ bias = 100$	US	$[-100, 100]^D$	$bias$
Shifted Rastrigin	$f_2 = \sum_{i=1}^{D} [y_i^2 - 10\cos(2\pi y_i) + 10] + bias, \quad y = x - o, \quad bias = 200$	MS	$[-5, 12, 5, 12]^D$	$bias$
Shifted Non Continuous Rastrigin	$f_3 = \sum_{i=1}^{D} [z_i^2 - 10\cos(2\pi z_i) + 10] + bias, \quad z = y - o, \quad y_i = \begin{cases} x_i & \text{if } \lvert x_i\rvert \le 0.5 \\ round(2x_i)/2, & \text{if } \lvert x_i\rvert > 0.5 \end{cases} bias = 300$	MS	$[-5, 12, 5, 12]^D$	$bias$
Shifted Discuss	$f_4 = 10^6 y_1^2 + \sum_{i=2}^{D} y_i^2 + bias,\ y = x - o,\ bias = 600$	US	$[-100, 100]^D$	$bias$
Shifted Levy	$f_5 = \sin^2(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2)[1 + 10\sin^2(\pi y_i + 1)] + (y_D - 1)^2[1 + \sin^2(2\pi y_D)] + bias, \quad y = x - o,\ bias = 500,$	MN	$[-50, 50]^D$	$biais$
Shifted Rotated H.C Elliptic	$f_6(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} y_i^2 + bias,\ y = M(x - o),\ bias = 400$	UN	$[-100, 100]^D$	$bias$
Shifted Rotated Rosenbrock	$f_7(x) = \sum_{i=1}^{D-1} (100(y_{i+1} - y_i^2)^2 + (y_i - 1)^2) + bias,\ y = M(x - o),\ bias = 700$	MN	$[-30, 30]^D$	$bias$
SR Expended Schaffer F6	$f_8 = g(y_1, y_2) + g(y_1, y_2) + \cdots + g(y_D, y_1) + bias,\ y = M(x - o),\ g(u, v) = 0.5 + \frac{\sin^2(u^2 + v^2) - 0.5}{(1 + 0.001(u^2 + v^2))^2},\ bias = 800,$	MN	$[-100, 100]^2$	$biais$
S.R. HappyCat	$f_9 = \lvert \sum_{i=1}^{D} y_i^2 - D\rvert^{\frac{1}{4}} + \frac{0.5}{D}(\sum_{i=1}^{D} y_i)^2 - \sum_{i=1}^{D} y_i) + 0.5,\ y = M(x - o)$	MN	$[-5, 10]^D$	$biais$
S.R. Zakharov	$f_{10} = \sum_{i=1}^{D} x_i^2 + (\sum_{i=1}^{D} 0.5 i x_i)^2 + (\sum_{i=1}^{D} 0.5 i x_i)^4 + biais\ y = M(x - o)\ biais = 1000$	UN	$[-5, 10]^D$	0
S.R. Ackley	$f_{11} = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=0}^{D} y_i^2}) - \exp(\frac{1}{D}\sum_{i=0}^{D} \cos(2\pi y_i) + 20 + e + biais,\ y = M(x - o),\ bias = 1100$	MN	$[-32.768, 32.768]^D$	$biais$
S.R HGBat	$f_{12} = \lvert(\sum_{i=1}^{D} x_i^2)^2 - (\sum_{i=1}^{D} x_i)^2\rvert^{0.5} + 0.5(\sum_{i=1}^{D} x_i)^2 - \sum_{i=1}^{D} x_i)/D + 0.5 + biais\ y = M(x - o)\ bias = 1200$	UN	$[-5, 10]^D$	$biais$
Quartic	$f_{13} = \sum_{i=1}^{D} i x_i^4 + rand(0, 1)$	MS	$[-1.28, 1.28]^D$	0
Inverted cosine wave	$f_{14} = -\sum_{i=1}^{D-1} \exp(-y_i/8)\cos(4\sqrt{y_i}),\ y_i = x_i^2 + x_{i+1}^2 + 0.5 x_i x_{i+1}$	MN	$[-5, 5]^D$	$-n + 1$
Penalized 1	$f_{15} = \frac{\pi}{D}\{10\sin^2(3\pi x_1) + \sum_{i=1}^{D-1} (y_i - 1)^2)[1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)^2\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)\ (*), \quad y_i = 1 + \frac{1}{4}(x_i + 1)$	MN	$[-50, 50]^D$	0
Himmelblau	$f_{16} = \frac{1}{D}\sum_{i=1}^{D}(x_i^4 - 16x_i^2 + 5x_i)$	MS	$[-5, 5]^D$	-78.3323
Alpine	$f_{17} = \sum_{i=1}^{D} \lvert x_i \sin(x_i) + 0.1 x_i\rvert$	MS	$[-10, 10]^D$	0
PowerSum	$f_{18} = \sum_{i=1}^{D} (\sum_{k=1}^{4} x_i^k - b_k)^2 \quad / \quad b = (8, 18, 44, 114)$	MN	$[0, n]^D$	0
Cosine Mixture	$f_{19} = \sum_{i=1}^{D} x_i^2 - 0.1 \sum_{i=1}^{D} cos(5\pi x_i)$	MS	$[-1; 1]^D$	$-0.1n$
Schwefel 2.22	$f_{20} = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$	UN	$[-10, 10]^D$	0
Powell sum	$f_{21} = \sum_{i=1}^{D} \lvert x_i\rvert^{i+1}$	MS	$[-100, 100]^D$	0
Easom	$f_{22} = -(-1)^D\big(\prod_{i=1}^{D} \cos(x_i)\big)\exp(-\sum_{i=1}^{D}(x_i - \pi)^2)$	UN	$[-10, 10]^D$	-1
Mishra 2	$f_{23} = (1 + \chi_D)^{\chi_D}, \quad \chi_D = D - \frac{1}{2}\sum_{i=1}^{D-1} x_i + x_{i+1}$	MN	$[0, 1]^D$	2
Brown	$f_{24} = \sum_{i=1}^{D-1} (x_i^2)^{(x_{i+1}^2 + 1)} + (x_{i+1}^2)^{(x_i^2 + 1)}$	UN	$[-1, 4]^D$	0

C: Characteristic, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable.

For the proposed Tornado algorithm, we have used the same set of values of the parameters (e.g. number of chaotic levels) for all experiments. The algorithm is not very sensitive to those parameters. In the current study, the parameters were set as follows:

- The number of CGS chaotic levels (N_c): $N_c = 5$.
- The number of CLS chaotic levels (N_l): $N_l = 5$.
- The number of CFS chaotic levels (N_f): $N_f = 10$.
- The number of CLS-CFS per cycle (M_l): $M_l = 100$.

4.1. Comparison with other Chaotic optimization algorithms

In order to show the effectiveness of our new chaotic optimization strategy, this section presents a comparison of the Tornado algorithm with state-of-the-art COA variants such as ICOLM [7] and ICOMM [24]. We have adopted the suggested parameters by the authors of those algorithms, as illustrated in Table 3. Three set of parameters have been suggested by the authors.

Configuration	M_g	M_l	M_{gl1}	M_{gl2}	λ	λ_{gl1}	λ_{gl2}
C1	800	400	6	6	0.1	0.04	0.01
C2	800	400	6	6	0.01	0.04	0.01
C3	800	400	6	6	0.001	0.04	0.01

Table 3: The set of parameters used by ICOLM and ICOMM approaches.

where M_g is the maximum number of iterations of chaotic global search, M_{gl1} is maximum number of iterations of first chaotic Local search in global search, $Mgl2$ is the maximum number of iterations of second chaotic local search in global search, M_l is the maximum number of iterations of chaotic local search, λ_{gl1} is the step size in first global–local search, λ_{gl2} is step size in second global–local search, and λ is the step size in chaotic local search. The other specific parameters of algorithms are given below:

- ICOLM uses Lozi map with: $a = 1.7,\ b = 0.5$.
- ICOMM uses Henon map with: $a = 4,\ b = 0.9$.

We choose the number of function evaluations (FEs) as a stopping criteria. The maximum number of function evaluations was 10^4 for all functions. Since the algorithms are stochastic in nature, 30 independent runs of each algorithm are carried out. The performance indicators used are the mean and the standard deviation. The comparison results for functions ($f_1 - f_{15}$) on moderate dimension $D = 10$ are shown in Table 4. It is observed from the results presented in Table 4 that the performance of our Tornado algorithm strongly dominates the existing COA approaches for all functions. Indeed, the computational results show clearly the deficiency of the classical COA approaches (here ICOLM and ICOMM) to even deal with moderate 10-dimensional problems whereas Tornado succeeds systematically. Therefore, it is needless to show the carried comparisons for large scale problems.

No	Stats	Tornado	ICOLM(1)	ICOLM(2)	ICOLM(3)	ICOMM(1)	ICOMM(2)	ICOMM(3)
F_1	Mean	5,30E-08	2,58E+08	2,28E+06	3,21E+04	2,44E+08	3,19E+06	8,98E+06
	Std	6,10E-08	6,23E+07	9,28E+05	1,13E+04	9,28E+07	3,66E+06	2,40E+07
F_2	Mean	1,24E+00	4,76E+02	2,99E+01	1,87E+03	1,36E+03	4,14E+02	1,04E+03
	Std	4,97E-01	9,73E+01	7,04E+00	7,64E+02	4,88E+02	9,11E+01	2,94E+02
F_3	Mean	2,75E+00	1,93E+01	7,05E+01	8,39E+01	2,82E+01	1,74E+01	2,21E+01
	Std	9,58E-01	8,05E+00	2,15E+01	2,18E+01	9,00E+00	9,14E+00	1,11E+01
F_4	Mean	5,62E-09	6,13E+02	9,92E+00	1,89E+04	1,99E+04	1,15E+04	8,33E+03
	Std	8,96E-09	4,47E+02	3,44E+00	9,36E+03	9,79E+03	3,26E+03	5,34E+03
F_5	Mean	0,00E+00	3,85E+00	3,31E+00	4,30E+00	4,99E-01	1,29E-01	1,26E-01
	Std	1,14E-13	9,93E-01	1,76E+00	1,40E+00	1,71E-01	9,85E-02	1,05E-01
F_6	Mean	3,14E-04	2,35E+05	3,91E+03	3,77E+01	7,84E+04	7,97E+03	2,35E+02
	Std	7,54E-04	1,51E+05	2,55E+03	1,89E+01	2,96E+04	9,05E+03	2,29E+02
F_7	Mean	8,51E+00	7,39E+01	1,82E+01	1,70E+01	1,93E+02	4,13E+01	1,21E+01
	Std	2,09E+01	4,28E+01	2,90E+01	2,89E+01	1,21E+02	3,22E+01	2,26E+01
F_8	Mean	5,10E-05	1,45E-01	4,46E-02	4,65E-03	8,78E-02	3,38E-02	6,78E-04
	Std	1,55E-04	1,01E-01	8,97E-02	8,41E-03	5,14E-02	3,39E-02	9,88E-04
F_9	Mean	1,99E+00	8,39E-01	8,80E-01	7,51E-01	1,04E+00	1,01E+00	9,08E-01
	Std	3,25E-02	1,99E-01	1,22E-01	2,24E-01	1,74E-01	1,93E-01	1,61E-01
F_{10}	Mean	8,87E+07	1,45E+04	8,59E+00	1,87E+05	9,09E+05	3,47E+05	9,42E+05
	Std	2,80E+08	2,88E+04	2,90E+00	1,51E+05	7,93E+05	3,83E+05	4,43E+05
F_{11}	Mean	2,00E+01	2,05E+01	2,06E+01	2,03E+01	2,04E+01	2,05E+01	2,04E+01
	Std	5,84E-03	7,42E-02	4,20E-02	1,03E-01	1,05E-01	9,15E-02	5,20E-01
F_{12}	Mean	4,87E-01	4,53E-01	4,75E-01	4,63E-01	4,92E-01	4,94E-01	4,99E-01
	Std	1,89E-02	3,36E-02	1,62E-02	1,58E-02	8,58E-03	6,69E-03	4,75E-03

Table 4: Comparison results for $f_1 - f_{12}$ on dimension $D = 10$ over 30 runs.

4.2. Comparison with state-of-the-art algorithms

We have also compared the obtained results with three state-of-the-art algorithms from different families of stochastic optimization algorithms:

- CMA-ES[3]: a Covariance Adaptation Evolution Strategy (ES) based algorithm [30]. It is a ES algorithm in which the Covariance matrix is deterministically adapted from the last move of the algorithm.
- L-SHADE[4]: SHADE is an adaptive Differential Evolution (DE) which incorporates success-history based parameter adaptation and one of the state-of-the-art DE algorithm. L-SHADE is an extension of SHADE using Linear Population Size Reduction (LPSR) [25].
- CLPSO[5]: a comprehensive learning particle swarm optimizer (CLPSO) embedded with local search (LS) is proposed to pursue higher optimization performance by taking the advantages of CLPSO's strong global search capability and LS's fast convergence ability [32].

The choice of the algorithms in the computational study is mainly driven by the high-quality of their results and the availability of code. We avoid the risk of non-optimal implementations and hence unfair comparisons. The maximum number of function evaluations is set to $2 \times 10^3 \times D$ for all algorithms and tested functions. We have adopted the suggested parameters by the authors of those algorithms. The computational results (i.e. error mean, standard deviation) are presented in Tables 5-7 for 30 independent runs. The obtained results show that the Tornado

[3] Available in the MATLAB library Yarpiz.
[4] Available at sites.google.com/site/tanaberyoji/software.
[5] Available in github.com/hmofrad/Adaptative-CLPSO.

algorithm dominated the other algorithms for most evaluated functions. The same conclusion has been observed for all dimensions of the functions (i.e. 50, 100, 200). For inherently separable benchmark functions and all dimensions (i.e. 50, 100 and 200), the Tornado algorithm is always better than the three other algorithms CLPPSO, CMA-ES and LSHADE. For the functions f_9 (multimodal, non-separable) and f_{10} (unimodal, non-separable), the CMA-ES (resp. CLPPSP) algorithm shows better performance for $D = 50$ ad $D = 100$ (resp. $D = 200$). For the shifted Levy function f_5 (multi-modal, non-separable), the algorithm LSHADE dominates the other algorithms.

No	CLPPSO		CMAES		Lshade		Tornado	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
f_1	7,17E+01	2,28E+01	2,44E+03	1,69E+03	3,48E−03	3,36E−03	**3,00E−12**	**2,46E−12**
f_2	2,51E+03	2,50E+03	3,23E+02	1,20E+01	3,60E+01	8,43E+00	**5,97E−01**	**1,33E+00**
f_3	**1,54E-03**	**1,22E-03**	2,82E+02	1,26E+01	1,11E+02	1,29E+01	2,20E+00	8,36E−01
f_4	1,97E+01	1,97E+01	2,75E+03	1,01E+03	1,00E−11	5,55E−12	**2,73E−12**	**4,40E−13**
f_5	1,87E-02	5,33E-03	9,80E−11	3,16E−11	**4,55E−13**	**1,14E−13**	1,24E−12	1,80E−13
f_6	7,42E-08	1,60E-08	3,33E−06	1,02E−06	9,00E−11	6,72E−11	**9,09E−13**	**1,80E−13**
f_7	4,89E+02	1,55E+01	3,86E+01	5,28E−01	4,38E+01	2,32E−02	**3,00E−02**	**3,16E−02**
f_8	4,08E-01	3,66E-02	1,00E−02	2,10E−03	5,00E−02	1,44E−02	**8,67E−02**	**9,17E−03**
f_9	3,44E+00	0,00E+00	**2,30E−01**	**2,97E−02**	2,95E+00	6,44E−05	2,95E+00	2,62E−04
f_{10}	1,93E+00	1,57E-01	**0,00E+00**	**1,51E−04**	0,00E+00	4,81E−04	2,70E−01	5,61E−02
f_{11}	2,13E+01	1,75E-02	2,12E+01	8,01E−02	2,09E+01	3,04E−01	**2,00E+01**	**1,16E−02**
f_{12}	4,92E-01	2,46E-03	4,90E−01	4,29E−02	**4,90E−01**	**1,12E−03**	5,00E−01	2,32E−03
f_{13}	3,85E-02	6,78E-03	6,51E−03	1,87E−03	3,15E−03	9,47E−04	**6,70E−04**	**3,96E−04**
f_{14}	1,24E+01	1,65E+00	1,37E+01	5,56E−01	3,55E−14	1,23E−14	**0,00E+00**	**1,07E−14**
f_{15}	3,49E-07	2,62E-07	1,81E−10	4,35E−11	2,68E−15	1,44E−15	**9,42E−33**	**0,00E+00**
f_{16}	1,51E-07	6,05E-08	1,47E+00	6,45E−01	9,00E−02	8,79E−02	**1,56E−09**	**3,26E−14**
f_{17}	1,42E-14	1,55E-14	1,22E−04	1,26E−04	**0,00E+00**	**0,00E+00**	**0,00E+00**	**0,00E+00**
f_{18}	3,36E-04	9,86E-05	3,03E−07	1,80E−07	2,96E−02	6,61E−02	**0,00E+00**	**0,00E+00**
f_{19}	2,66E+00	2,74E-02	2,80E+00	1,57E+00	2,56E+00	1,43E+00	**2,04E−15**	**5,96E−16**
f_{20}	8,68E+00	2,45E+00	2,21E−04	6,51E−05	3,26E−06	1,92E−06	**6,04E−170**	**0,00E+00**
f_{21}	2,94E-04	1,51E-05	4,44E+10	7,46E+10	3,32E−16	5,81E−16	**2,52E−196**	**0,00E+00**
f_{22}	1,00E+00	0,00E+00	1,00E+00	0,00E+00	0,00E+00	0,00E+00	**0,00E+00**	**9,61E−17**
f_{23}	0,00E+00	0,00E+00	1,33E+00	3,18E−08	4,00E−08	1,87E−08	**0,00E+00**	**0,00E+00**
f_{24}	2,54E-07	1,30E-07	2,32E−11	1,09E−11	6,59E−15	5,40E−15	**4,99E−77**	**2,16E−77**

Table 5: Comparison results for $f_1 - f_{24}$ problems on dimension $D = 50$ over 30 runs.

The final ranking of the evaluated algorithms is performed by using all the obtained results. The algorithms are sorted for each test function. The ranking are summed up and are presented in Tables 8-10. Clearly the Tornado algorithm is the winner, while LSHADE is the runner-up. We notice also that the performance of the CMA-ES algorithm decreases function of the dimension of the problem.

No	CLPPSO		CMAES		Lshade		Tornado	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
F_1	1,86E+06	4,33E+05	1,10E+01	8,96E+00	1,10E+01	8,96E+00	**6,37E−12**	**2,96E−12**
F_2	1,38E+02	1,13E+01	1,97E+02	2,19E+01	1,97E+02	2,19E+01	**1,10E+00**	**4,22E−01**
F_3	2,69E+01	1,27E+01	1,27E+02	7,72E+00	1,27E+02	7,72E+00	**5,61E+00**	**2,10E+00**
F_4	1,63E+00	3,66E-01	3,95E−08	2,34E−08	3,95E−08	2,34E−08	**4,32E−12**	**3,94E−13**
F_5	2,15E+02	4,34E+01	7,92E−07	5,44E−07	7,92E−07	5,44E−07	**1,59E−12**	**2,60E−13**
F_6	1,84E-03	2,24E-04	3,25E−11	1,56E−11	3,25E−11	1,56E−11	**1,59E−12**	**1,97E−13**
F_7	4,89E+02	1,55E+01	8,70E+01	9,60E−01	1,22E+02	1,75E+00	**2,58E−02**	**8,84E−03**
F_8	4,08E-01	3,66E-02	3,45E−03	7,91E−04	2,08E−02	3,51E−03	**1,56E−04**	**3,77E−04**
F_9	3,44E+00	0,00E+00	**2,00E−01**	**2,42E−02**	3,44E+00	2,51E−04	3,44E+00	9,72E−04
F_{10}	1,93E+00	1,57E-01	**1,76E−05**	**7,07E−06**	4,29E−03	1,09E−03	9,48E−01	1,12E−01
F_{11}	2,13E+01	1,75E-02	2,13E+01	1,93E−02	2,13E+01	3,64E−02	**1,98E+01**	**4,31E−02**
F_{12}	4,92E-01	2,46E-03	5,04E−01	3,16E−02	4,92E−01	1,40E−03	**5,00E−01**	**3,73E−02**
F_{13}	5,86E-02	1,02E-02	1,37E−02	2,18E−03	7,72E−03	2,03E−03	**5,40E−04**	**5,52E−05**
F_{14}	3,04E+01	4,94E-01	2,82E+01	1,99E+00	1,33E−11	4,26E−12	**0,00E+00**	**2,25E−14**
F_{15}	5,52E-08	2,69E-08	4,39E−12	1,86E−12	3,54E−12	2,16E−12	**4,71E−33**	**0,00E+00**
F_{16}	4,79E-08	1,47E-08	2,83E+00	1,26E+00	5,35E+00	3,54E−01	**1,41E−06**	**1,00E−14**
F_{17}	1,88E-17	1,59E-17	8,46E−05	8,67E−05	2,02E−28	1,24E−28	**0,00E+00**	**0,00E+00**
F_{18}	1,03E-04	2,37E-05	5,10E−09	1,89E−09	7,98E−01	2,88E−01	**0,00E+00**	**0,00E+00**
F_{19}	3,47E+00	3,12E-02	3,50E+00	8,60E−16	3,50E+00	1,25E−11	**5,33E−15**	**1,54E−15**
F_{20}	9,26E+00	5,79E+00	3,35E−03	8,92E−04	3,02E−01	7,29E−02	**4,20E+00**	**1,59E+00**
F_{21}	2,23E-04	2,59E-05	4,12E−05	7,78E−06	4,56E−04	3,56E−04	**0,00E+00**	**0,00E+00**
F_{22}	1,00E+00	0,00E+00	1,00E+00	0,00E+00	1,00E+00	0,00E+00	**9,99E−16**	**1,57E−16**
F_{23}	0,00E+00	0,00E+00	9,77E−01	2,33E+01	8,53E−05	3,07E−05	**0,00E+00**	**0,00E+00**
F_{24}	2,49E-07	7,74E-08	1,63E−12	9,08E−13	1,47E−10	1,02E−10	**4,99E−148**	**4,20E−148**

Table 6: Comparison results for $f_1 - f_{24}$ problems on dimension $D = 100$ over 30 runs.

No	CLPPSO		CMAES		Lshade		Tornado	
	Mean	**Std**	**Mean**	**Std**	**Mean**	**Std**	**Mean**	**Std**
F_1	2,44E+03	1,69E+03	1,19E+00	3,52E-01	3,48E-03	3,36E-03	**3,00E-12**	**2,46E-12**
F_2	3,23E+02	1,20E+01	7,86E+01	5,67E+00	3,60E+01	8,43E+00	**5,97E-01**	**1,33E+00**
F_3	2,82E+02	1,26E+01	2,38E+01	4,97E+00	1,11E+02	1,29E+01	**2,20E+00**	**8,36E-01**
F_4	2,75E+03	1,01E+03	1,04E+04	4,40E+03	1,00E-11	5,55E-12	**2,73E-12**	**4,40E-13**
F_5	9,80E-11	3,16E-11	**0,00E+00**	**0,00E+00**	4,55E-13	1,14E-13	1,24E-12	1,80E-13
F_6	3,33E-06	1,02E-06	4,52E-08	9,63E-09	9,00E-11	6,72E-11	**9,09E-13**	**1,80E-13**
F_7	3,86E+01	5,28E-01	1,85E+02	3,90E-01	4,38E+01	2,32E-02	**3,00E-02**	**3,16E-02**
F_8	1,00E-02	2,10E-03	1,00E-05	4,99E-06	5,00E-02	1,44E-02	**8,67E-02**	**9,17E-03**
F_9	**2,30E-01**	**2,97E-02**	2,77E-01	2,64E-02	2,95E+00	6,44E-05	2,95E+00	2,62E-04
F_{10}	**0,00E+00**	**1,51E-04**	4,09E+03	6,24E+03	0,00E+00	4,81E-04	2,70E-01	5,61E-02
F_{11}	2,12E+01	8,01E-02	2,15E+01	1,03E-02	2,09E+01	3,04E-01	**2,00E+01**	**1,16E-02**
F_{12}	4,90E-01	4,29E-02	5,08E-01	5,40E-02	**4,90E-01**	**1,12E-03**	5,00E-01	2,32E-03
F_{13}	6,51E-03	1,87E-03	2,92E-02	2,21E-03	3,15E-03	9,47E-04	**6,70E-04**	**3,96E-04**
F_{14}	1,37E+01	5,56E-01	6,12E+01	1,03E+01	3,55E-14	1,23E-14	**0,00E+00**	**1,07E-14**
F_{15}	1,81E-10	4,35E-11	2,87E-15	1,54E-16	2,68E-15	1,44E-15	**9,42E-33**	**0,00E+00**
F_{16}	1,47E+00	6,45E-01	3,46E+00	1,30E+00	9,00E-02	8,79E-02	**1,56E-09**	**3,26E-14**
F_{17}	1,22E-04	1,26E-04	**0,00E+00**	**0,00E+00**	**0,00E+00**	**0,00E+00**	**0,00E+00**	**0,00E+00**
F_{18}	3,03E-07	1,80E-07	1,26E-12	2,71E-14	2,96E-02	6,61E-02	**0,00E+00**	**0,00E+00**
F_{19}	2,80E+00	1,57E+00	3,50E+00	0,00E+00	2,56E+00	1,43E+00	**2,04E-15**	**5,96E-16**
F_{20}	2,21E-04	6,51E-05	2,14E-03	4,75E-04	3,26E-06	1,92E-06	**6,04E-170**	**0,00E+00**
F_{21}	4,44E+10	7,46E+10	1,23E-06	9,76E-08	3,32E-16	5,81E-16	**2,52E-196**	**0,00E+00**
F_{22}	1,00E+00	0,00E+00	1,00E+00	0,00E+00	0,00E+00	0,00E+00	**0,00E+00**	**9,61E-17**
F_{23}	1,33E+00	3,18E-08	1,33E+02	1,54E+02	4,00E-08	1,87E-08	**0,00E+00**	**0,00E+00**
F_{24}	2,32E-11	1,09E-11	3,63E-16	2,00E-16	6,59E-15	5,40E-15	**4,99E-77**	**2,16E-77**

Table 7: Comparison results for $f_1 - f_{24}$ problems on dimension $D = 200$ over 30 runs.

Table 8: The rank of the four algorithms for the functions test on $D = 50$.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}	Mean Rank
CLLPSO	4	4	1	3	4	3	4	4	4	4	4	4	4	3	4	2	4	3	4	3	4	3	3	4	3,52
CMAES	3	2	4	4	3	4	2	1	1	2	3	2	3	4	3	4	1	2	3	4	3	4	4	2	2,76
Lshade	2	3	3	2	1	2	3	2	2	3	2	1	2	2	2	3	1	4	2	1	2	1	2	3	2,12
Tornado	1	1	2	1	2	1	1	3	3	1	1	3	1	1	1	1	1	1	1	2	1	2	1	1	**1,48**

Table 9: The rank of the four algorithms for the functions test on $D = 100$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}	Mean Rank
CLLPSO	4	2	4	4	4	4	3	4	4	4	2	2	3	3	4	3	4	3	3	3	4	3	3	4	3,40
CMAES	3	3	3	3	3	2	1	1	1	2	3	3	4	4	3	4	1	2	4	4	3	4	4	2	2,72
Lshade	2	4	2	2	2	3	2	2	2	3	4	1	2	2	2	2	1	4	2	2	1	1	2	3	2,20
Tornado	1	1	1	1	1	1	4	3	3	1	1	4	1	1	1	1	1	1	1	1	2	2	1	1	**1,56**

Table 10: The rank of the for algorithms four the functions test on $D = 200$

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}	Mean Rank
CLLPSO	4	1	4	3	4	4	2	2	1	1	3	2	3	3	4	3	4	3	4	3	4	3	3	4	3,04
CMAES	3	4	3	4	1	3	4	1	2	4	4	4	4	4	3	4	1	2	3	4	3	4	4	2	3,12
Lshade	2	3	2	2	2	2	3	3	3	2	2	1	2	2	2	2	1	4	2	1	2	1	2	3	2,12
Tornado	1	2	1	1	3	1	1	4	4	3	1	3	1	1	1	1	1	1	1	2	1	2	1	1	**1,62**

Table 11: Total of Mean time (per run) consumed by the for algorithms on dimensions $D = 50$, $D = 100$ and $D = 200$.

	CLLPSO	Lshade	CMAES	Tornado
Total of mean Time for $D = 50$	247,2 s	95,1 s	902,2 s	109,2 s
Total of mean Time for $D = 100$	536,4 s	328,3 s	2640,3 s	215,1 s
Total of mean Time for $D = 200$	1415,4 s	1385,4 s	10425,6 s	614,8 s

Table11 shows the execution time of the algorithms for the same number of evaluated objective functions. Clearly, the Tornado algorithm is the fatest one, while CMA-ES is the slowest one. Indeed, at each iteration the CMA-ES algorithm computes the covariance matrix which is a time-consuming task. Figure shows the convergence of the algorithms for functions $f_1 - f_{10}$ on dimension $D = 50$ and $D = 100$. The obtained results show a fast convergence for the Tornado algorithm for most of the functions. Other carried experiments show the same trend for problems with $D = 200$.

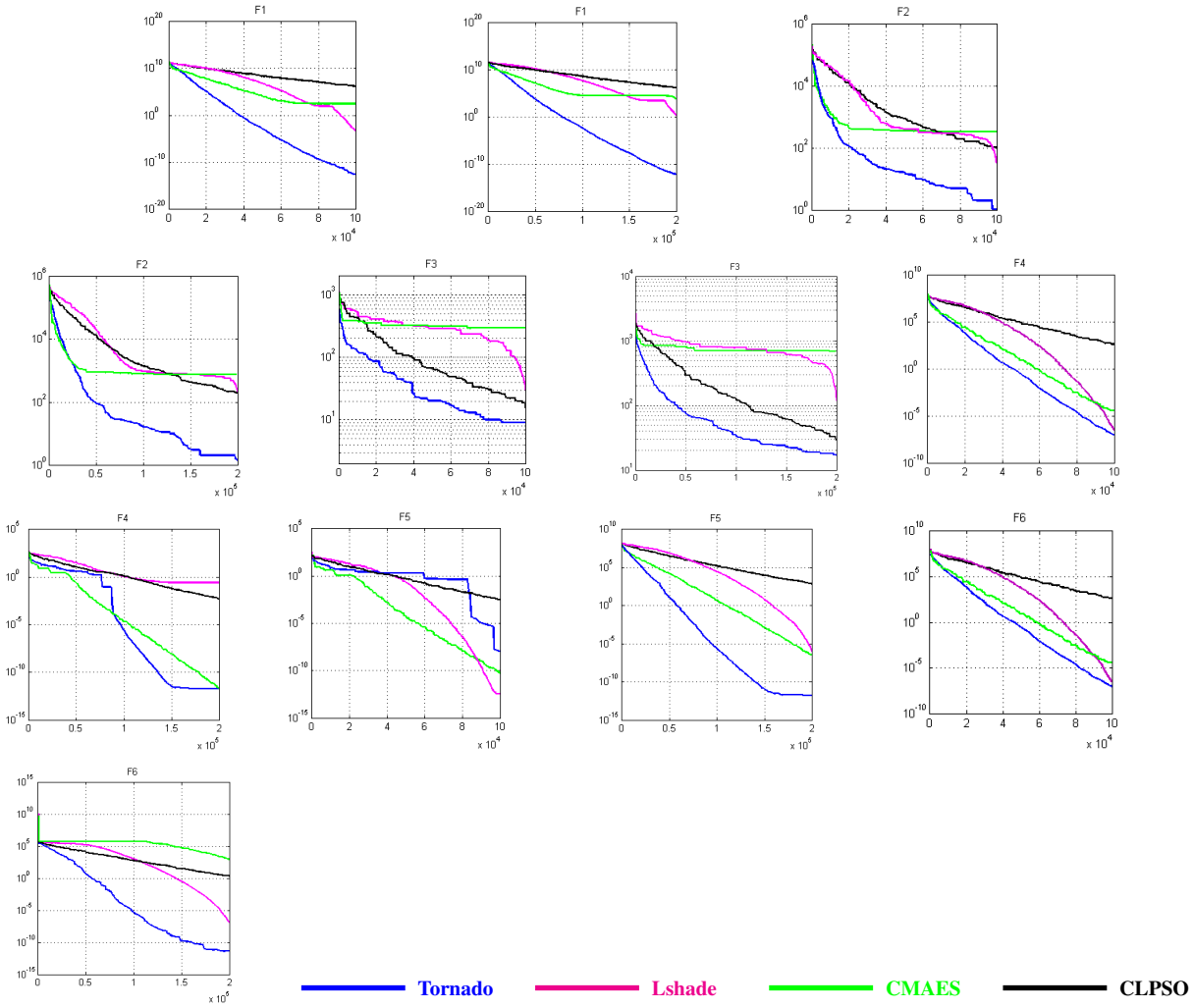


Figure 14: Convergence performance of the four different methods for functions $f_1 - f_6$ on dimensions $D = 50$ (left) and $D = 100$ (right).

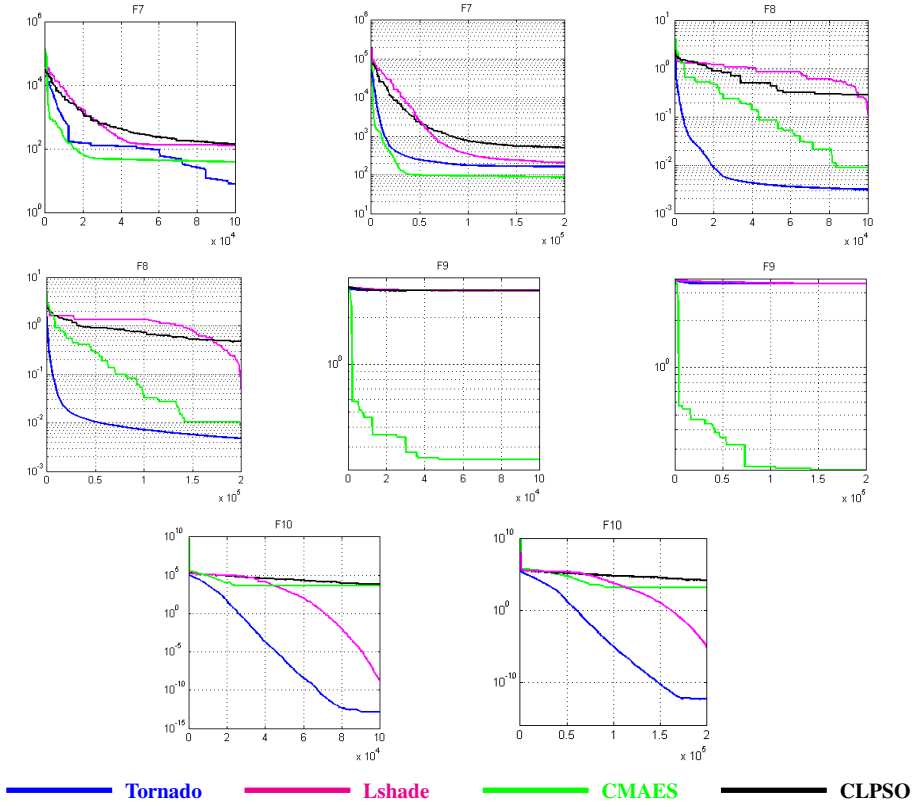


Figure 15: Convergence performance of the four different methods for functions $f_7 - f_{10}$ on dimensions $D = 50$ (left) and $D = 100$ (right).

5. Conclusions and perspectives

In the big era, there is a need for developing optimization algorithms able to effectively solve problems with hundreds, thousands, and even millions of variables. In this paper we have proposed an autonomous chaotic optimization algorithm, called Tornado, for large scale global optimization problems. The algorithm introduces advanced symmetrization, levelling and fine search strategies for an efficient and effective exploration of the search space and exploitation of the best found solutions. To our knowledge, this is the first accurate and fast autonomous chaotic algorithm solving large scale optimization problems.

The obtained results has shown the scalability of the algorithm in contrast to chaotic optimization algorithms encountered in the literature. Moreover, in comparison with some state-of-the-art metaheuristics (e.g. evolutionary algorithms, swarm intelligence), the computational results revealed that the proposed Tornado algorithm is an effective and efficient optimization algorithm. The Tornado algorithm shows promising capabilities in balancing the exploitation and the exploration of the search space.

We will investigate the application of the Tornado algorithm to large scale real-life optimization problems such as learning of deep neural networks, the optimization of the hyper-parameters of deep convolution neural networks, and demand energy management in smart grids. An extension of the Tornado algorithm to solve multi-objective optimization problems using scalarization and Pareto approaches is also under study.

A parallel implementation of the algorithm on heterogeneous parallel architectures, composed of multi-cores and clusters of GPUs, will be also investigated. We are also interested in the design of Fractals based decomposition strategies. The Chaotic approach will be combined to a Fractal based decomposition model, in which chaotic search is applied in each Fractal. This combination will generate highly parallel approaches to be implemented on exascale

parallel architectures composed of millions of GPU cores. The parallel model will also improve the exploration capabilities of the Tornado algorithm.

References

[1] X. Wu and Z. Chen, "Introduction of chaos theory, shanghai science and technology," 1996.

[2] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of the atmospheric sciences*, vol. 20, no. 2, pp. 130–141, 1963.

[3] D. Auerbach, C. Grebogi, E. Ott, and J. A. Yorke, "Controlling chaos in high dimensional systems," *Physical review letters*, vol. 69, no. 24, p. 3479, 1992.

[4] L. M. Pecora and T. L. Carroll, "Synchronization in chaotic systems," *Physical review letters*, vol. 64, no. 8, p. 821, 1990.

[5] B. Li and W. Jiang, "Chaos optimization method and its application," *Control Theory & Applications*, vol. 4, 1997.

[6] D. Yang, G. Li, and G. Cheng, "On the efficiency of chaos optimization algorithms for global optimization," *Chaos, Solitons & Fractals*, vol. 34, no. 4, pp. 1366–1375, 2007.

[7] T. Hamaizia and R. Lozi, "Improving chaotic optimization algorithm using a new global locally averaged strategy," in *Emergent Properties in Natural and Artificial Complex Systems*, pp. pp–17, 2011.

[8] N. Aslimani and R. Ellaia, "A new chaos optimization algorithm based on symmetrization and levelling approaches for global optimization," *Numerical Algorithms*, vol. 79, no. 4, pp. 1021–1047, 2018.

[9] J. Feng, J. Zhang, X. Zhu, and W. Lian, "A novel chaos optimization algorithm," *Multimedia Tools and Applications*, vol. 76, no. 16, pp. 17405–17436, 2017.

[10] L. Shengsong, W. Min, and H. Zhijian, "Hybrid algorithm of chaos optimisation and slp for optimal power flow problems with multimodal characteristic," *IEE Proceedings-Generation, Transmission and Distribution*, vol. 150, no. 5, pp. 543–547, 2003.

[11] J. Wang and X. Wang, "A global control of polynomial chaotic systems," *International Journal of Control*, vol. 72, no. 10, pp. 911–918, 1999.

[12] S. Ishii and M.-a. Sato, "Constrained neural approaches to quadratic assignment problems," *Neural Networks*, vol. 11, no. 6, pp. 1073–1082, 1998.

[13] K.-w. Wong, K.-P. Man, S. Li, and X. Liao, "A more secure chaotic cryptographic scheme based on the dynamic look-up table," *Circuits, Systems and Signal Processing*, vol. 24, no. 5, pp. 571–584, 2005.

[14] H. Gao, Y. Zhang, S. Liang, and D. Li, "A new chaotic algorithm for image encryption," *Chaos, Solitons & Fractals*, vol. 29, no. 2, pp. 393–399, 2006.

[15] R. A. Ibrahim, M. A. Elaziz, and S. Lu, "Chaotic opposition-based grey-wolf optimization algorithm based on differential evolution and disruption operator for global optimization," *Expert Systems with Applications*, vol. 108, pp. 1–27, 2018.

[16] J. A. Koupaei, S. M. M. Hosseini, and F. M. Ghaini, "A new optimization algorithm based on chaotic maps and golden section search method," *Engineering Applications of Artificial Intelligence*, vol. 50, pp. 201–214, 2016.

[17] S. Arora and S. Singh, "An improved butterfly optimization algorithm with chaos," *Journal of Intelligent & Fuzzy Systems*, vol. 32, no. 1, pp. 1079–1088, 2017.

[18] M. Petrović, N. Vuković, M. Mitić, and Z. Miljković, "Integration of process planning and scheduling using chaotic particle swarm optimization algorithm," *Expert systems with Applications*, vol. 64, pp. 569–588, 2016.

[19] L. Wang, X. Liu, M. Sun, J. Qu, and Y. Wei, "A new chaotic starling particle swarm optimization algorithm for clustering problems," *Mathematical Problems in Engineering*, vol. 2018, 2018.

[20] G.-G. Wang, S. Deb, A. H. Gandomi, Z. Zhang, and A. H. Alavi, "Chaotic cuckoo search," *Soft Computing*, vol. 20, no. 9, pp. 3349–3362, 2016.

[21] I. Fister Jr, M. Perc, S. M. Kamal, and I. Fister, "A review of chaos-based firefly algorithms: perspectives and research challenges," *Applied Mathematics and Computation*, vol. 252, pp. 155–165, 2015.

[22] H. Yan, L. Zhou, and L. Liu, "Chaos genetic algorithm optimization design based on permanent magnet brushless dc motor," in *Proceedings of the 2015 International Conference on Electrical and Information Technologies for Rail Transportation*, pp. 329–337, Springer, 2016.

[23] L. dos Santos Coelho, "Tuning of pid controller for an automatic regulator voltage system using chaotic optimization approach," *Chaos, Solitons & Fractals*, vol. 39, no. 4, pp. 1504–1514, 2009.

[24] T. Hamaizia, R. Lozi, and N.-e. Hamri, "Fast chaotic optimization algorithm based on locally averaged strategy and multifold chaotic attractor," *Applied Mathematics and Computation*, vol. 219, no. 1, pp. 188–196, 2012.

[25] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *2014 IEEE congress on evolutionary computation (CEC)*, pp. 1658–1665, IEEE, 2014.

[26] J. Brest, M. S. Maučec, and B. Bošković, "Single objective real-parameter optimization: Algorithm jso," in *2017 IEEE congress on evolutionary computation (CEC)*, pp. 1311–1318, IEEE, 2017.

[27] R. Salgotra, U. Singh, and G. Singh, "Improving the adaptive properties of lshade algorithm for global optimization," in *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*, pp. 400–407, IEEE, 2019.

[28] R. Salgotra, U. Singh, S. Saha, and A. Nagar, "New improved salshade-cnepsin algorithm with adaptive parameters," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 3150–3156, IEEE, 2019.

[29] S. Akhmedova, V. Stanovov, and E. Semenkin, "Lshade algorithm with a rank-based selective pressure strategy for the circular antenna array design problem.," in *ICINCO (1)*, pp. 159–165, 2018.

[30] I. Loshchilov, "Cma-es with restarts for solving cec 2013 benchmark problems," in *2013 IEEE Congress on Evolutionary Computation*, pp. 369–376, Ieee, 2013.

[31] L. T. Al-Bahrani and J. C. Patra, "A novel orthogonal pso algorithm based on orthogonal diagonalization," *Swarm and Evolutionary Computation*, vol. 40, pp. 1–23, 2018.

[32] N. Lynn and P. N. Suganthan, "Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation," *Swarm and Evolutionary Computation*, vol. 24, pp. 11–24, 2015.

[33] P. Larrañaga and J. A. Lozano, *Estimation of distribution algorithms: A new tool for evolutionary computation*, vol. 2. Springer Science & Business Media, 2001.
[34] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm and evolutionary computation*, vol. 1, no. 3, pp. 111–128, 2011.
[35] A. Kabán, J. Bootkrajang, and R. J. Durrant, "Toward large-scale continuous eda: A random matrix theory perspective," *Evolutionary computation*, vol. 24, no. 2, pp. 255–291, 2016.
[36] W. Dong, T. Chen, P. Tiňo, and X. Yao, "Scaling up estimation of distribution algorithms for continuous optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 6, pp. 797–822, 2013.
[37] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña, "Optimization in continuous domains by learning and simulation of gaussian networks," 2000.
[38] A. W. Mohamed, A. A. Hadi, A. M. Fattouh, and K. M. Jambi, "Lshade with semi-parameter adaptation hybrid with cma-es for solving cec 2017 benchmark problems," in *2017 IEEE Congress on evolutionary computation (CEC)*, pp. 145–152, IEEE, 2017.
[39] G. Zhang and Y. Shi, "Hybrid sampling evolution strategy for solving single objective bound constrained problems," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–7, IEEE, 2018.